Advanced Topics in Communications II – ELCN456

GOS Calculator Report

Submitted To:

Dr. Samy Soliman

Eng. Menna

Submitted By:

Loay Samy – 1170031

Kareem Mohamed – 4180299

Marwan Ehab – 1170057

Table of Contents

List of figures

1. Task Definition

Design and implement a software-based GUI Grade of Service (GOS) Calculator using MATLAB based on **Erlang B Formula**, **Erlang C Formula** and **Binomial Formula**.

The Gos Calculator has two parts:

- Part 1

Given the number of trunks N, K users, average call rate λ calls/hour, and call holding time H min/call It calculates the value of traffic intensity A in Erlangs, and Gos percentage.

- Part 2

Given Gos in percentage, N trunks and K in case we are using Binomial formula, It Calculates the corresponding total traffic offered to this group in erlangs.

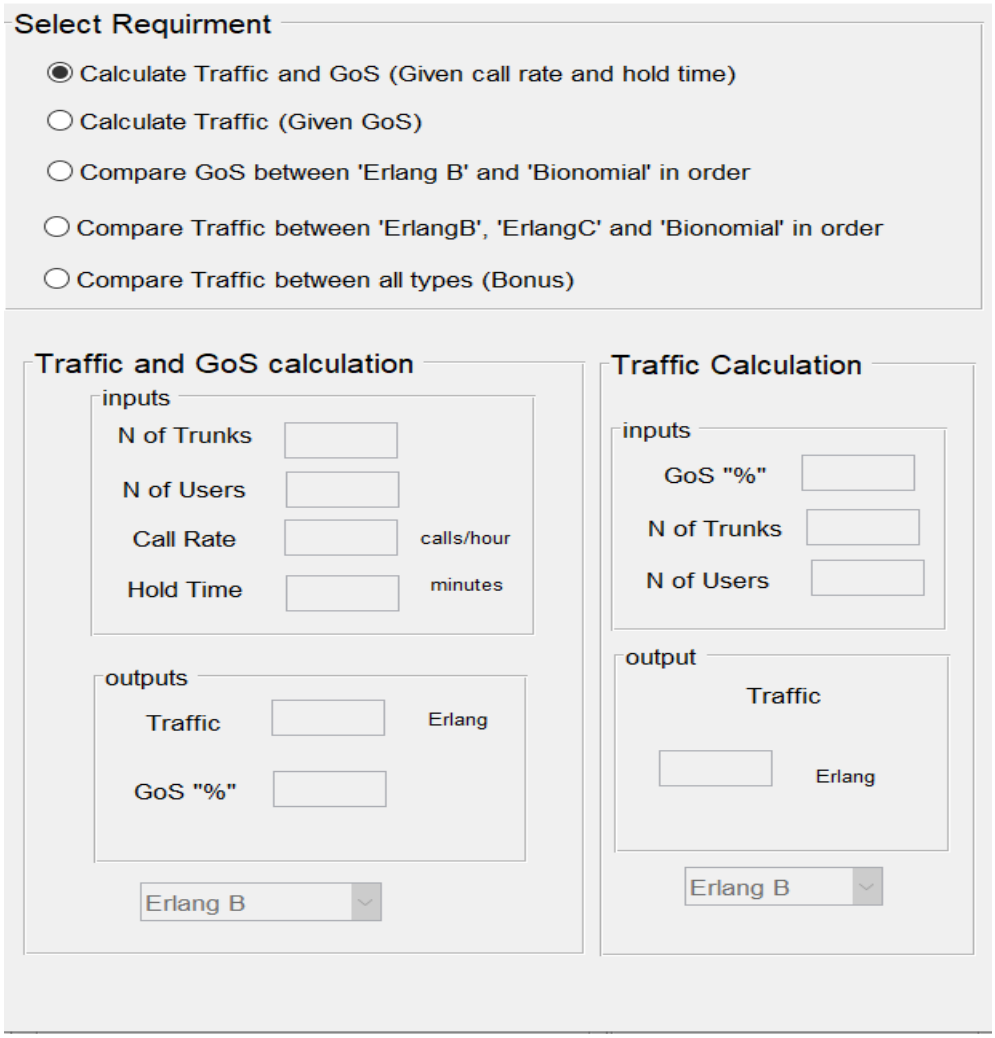


Figure 1 Calculator Parameters

2. Functions

2.1 Erlang B

We make two functions for ErlangB, one for part 1 which takes number of trunks N, K users, average call rate λ calls/hour, and call holding time H min/call It calculates the value of traffic intensity A in Erlangs, and Gos percentage.

```
function [A,Gos] = Erlang_B_Part1(N,K,Y,H)
if(K>N)
Au = (H * Y)/60 ;
A = K * Au ;
ErlangB = @(A) (A^N/factorial(N))/sum(A.^([0:N])./cumprod([0,0:N-1]+1));
Gos = ErlangB(A);
Gos = Gos * 100;
else
    Gos = 0;
    Au = (H * Y)/60 ;
    A = K * Au ;
end

end
```

Figure 2 Erlang B Part 1

Notes: grade of service will be equal zero in case that the number of trunks is more than number of users since probability of blocking will be zero.

The second function Given Gos in percentage, N trunks, It Calculates the corresponding total traffic A offered to this group in erlangs.

```
function [A] = Erlang_B_Part2(N,Gos)
Gos = Gos/100;
ErlangB = @(A) (A^N/factorial(N))/sum(A.^([0:N])./cumprod([0,0:N-1]+1));
A = fsolve(@(A) ErlangB(A)-Gos, N);
disp(A);
end
```

Figure 3 Erlang B Part2

2.2 Binomial

We have two functions too for Binomial formula one for part1, which takes number of trunks **N, K** users, average call rate **λ** calls/hour, and call holding time **H** min/call It calculates the value of **traffic intensity A** in Erlangs, and **Gos percentage**

```
function [A,Gos] = Erlang_Bino_Part1(N,K,Y,H)
Au = (H * Y)/60 ;
A = K * Au ;
if(K>N)
    if(Au > 1)
        Gos = 100;
    else

index = 1;
Compute=zeros(1,K-N-1);
for i=N:K-1
    Compute(index)= nchoosek(K-1,i);
    index = index+1;
end
disp(Compute);
Y=Compute.*Au.^([N:K-1]).*(1-Au).^(K-1.-[N:K-1]);
disp(Y);
ErlangBino = @(Au)(sum(Compute.*Au.^([N:K-1]).*(1-Au).^(K-1.-[N:K-1]))); %% Erlang Bino equation
Gos = ErlangBino(Au);
Gos = Gos*100;
    end
else
    Gos =0;
end
end
```

Figure 4 Binomial function part 1

Notes: here we make if condition for Au which is bigger than one the grade of service for such case will always be 100 since if we have a calling rate 3 calls/hr and the holding time is 1 hr/call so it will always happen blocking for the calls.

The second function Given **Gos** in percentage, **N** trunk, K Number of users, It Calculates the corresponding total traffic **A** offered to this group in erlangs

```
function [A] = Erlang_Binomial_Part2(N,Gos,K)
Gos=Gos/100;
if(K>N)
Compute=zeros(1,K-N-1);
index= 1;
for i=N:K-1
    Compute(index)= nchoosek(K-1,i);
    index = index+1;
end
ErlangBino = @(Au)(sum(Compute.*Au.^([N:K-1]).*(1-Au).^(K-1.-[N:K-1]))); %% Erlang Bino equation
itterator=N;
Au = fsolve(@(Au) ErlangBino(Au)-Gos, N);
while(Au < 0)
    itterator=itterator/2;
    Au = fsolve(@(Au) ErlangBino(Au)-Gos,itterator);
end
disp(Au);
A = Au*K;
disp(A);
end
end
```

Figure 5 Binomial function part 1

2.3 Erlang C

We have two functions too for ErlangC formula one for part1, which takes number of trunks N, K users, average call rate λ calls/hour, and call holding time H min/call It calculates the value of traffic intensity A in Erlangs, and Gos percentage.

```
function [A,Gos] = Erlang_C_Part1(N,K,Y,H)
Au = (H * Y)/60 ;
A = K * Au ;
if(K>N)
ErlangC = @(A)((N*A^(N)/(factorial(N)*(N-A)))/((N*A^(N)/(factorial(N)*(N-A)))+sum(A.^([0:N-1])./cumprod([0,0:N-2]+1))));
Gos = ErlangC(A);
Gos = Gos * 100;
else
    Gos =0;
end
end
```

Figure 6 Erlang C function part 1

The second function Given Gos in percentage, N trunks, It Calculates the corresponding total traffic A offered to this group in erlangs.

```
function [A] = Erlang_C_Part2(N,Gos)
Gos = Gos/100;
ErlangC = @(A)((N*A^(N)/(factorial(N)*(N-A)))/((N*A^(N)/(factorial(N)*(N-A)))+sum(A.^([0:N-1])./cumprod([0,0:N-2]+1)))); %% Erlang C equation
A = fsolve(@(A) ErlangC(A)-Gos, N+1);
disp(A);
end
```

Figure 7 Erlang C function part 2

3. GUI Implementation

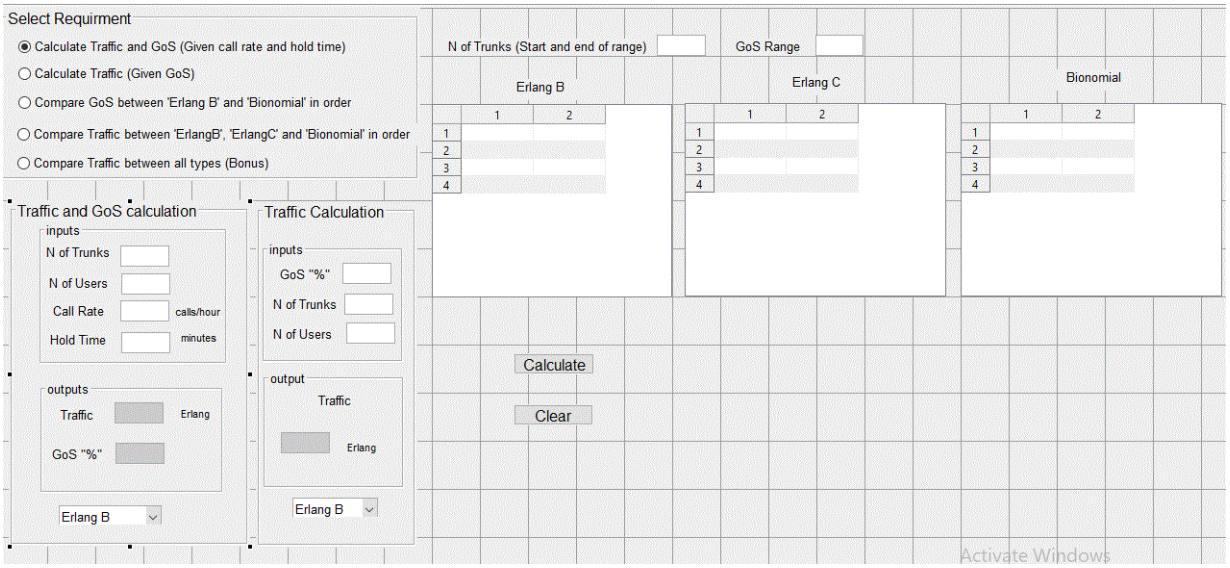


Figure 8 GUI

First guide in the GUI is to select the requirement from the 'Select Requirement' panel as initially all the text boxes are disable until one of the following requirement is taken:

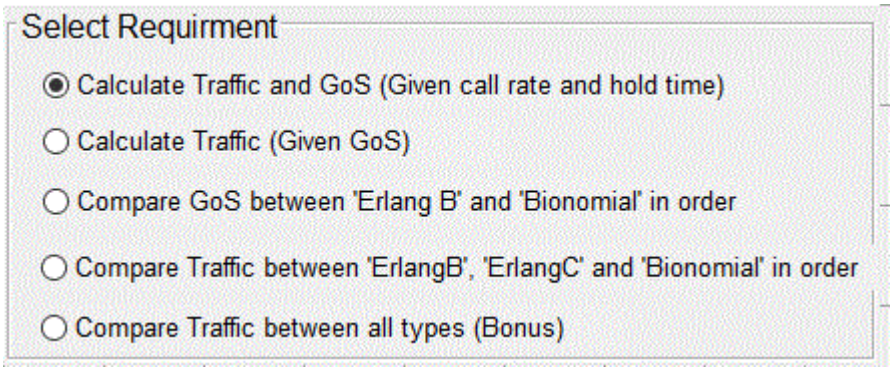


Figure 9 Select Requirement

Selecting one of them enables its corresponding instance in the GUI detailed as follows in order:

3.1 GoS calculator:

Part a: (Calculate traffic and GoS)

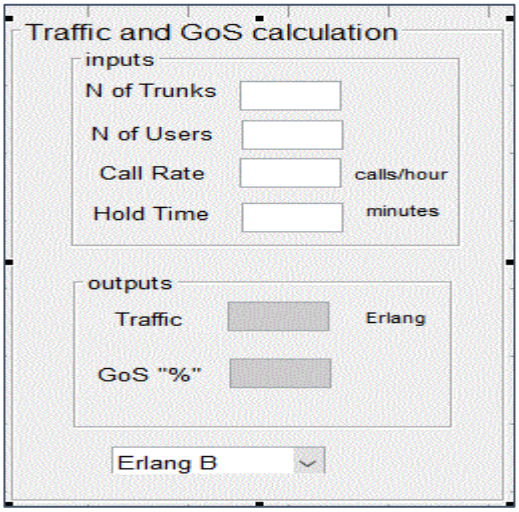


Figure 10 Part a: (Calculate traffic and GoS)

GUI takes number of trunks, number of users, call rate and hold time from the user as text input, the user also selects which GoS method is required within Erlang, ErlangC and Binomial from the popup menu. Based on these inputs, traffic and GoS are calculated according to their formula in the functions part mentioned previously then displayed in their text boxes in the GUI.

Part b: (Calculate traffic (given GoS))

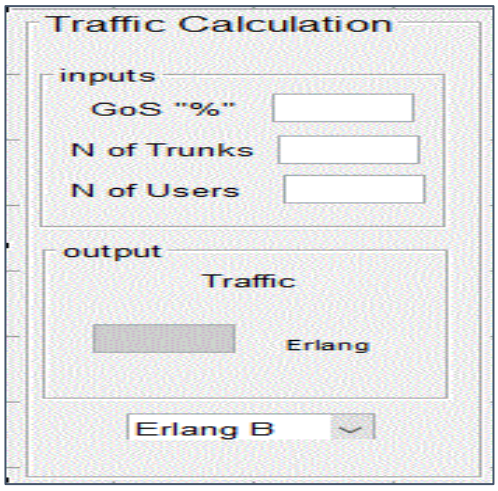


Figure 11 Part b: (Calculate traffic (given GoS))

GUI takes number of trunks and GoS, the user also selects which GoS method is required within Erlang, ErlangC and Binomial from the popup menu. Based on these inputs, traffic is calculated according to its formula in the functions part mentioned previously then displayed in its text box in the GUI. Selecting Binomial formula is an exception as it requires the user also to enter the number of users 'K' parameter as its needed in this formula.

3.2 GoS comparator:

This part compares the GoS of ErlangB and Binomial for call rate 'λ' = 3 calls/hour, hold time 'H' = 4 minutes/call and all combination of number of users 'K' and number of trunks 'N'. 'K' ranges from 5-50 with 5 increments and 'N' with a range from 1 to 10 with unity increment. The table is generated in a new figure with 'N' on the column axes, 'K' on the row axes and each column contains two internal columns of the two types; ErlangB and Binomial in order for better illustration of the compared GoS values.

3.3 Traffic comparator:

This part compares the GoS of ErlangB, ErlangC and Binomial for all combination of GoS and number of trunks 'N'. 'GoS' values are (0.5%, 1%, 2%, 3%, 5%) and 'N' with a range from 1 to 10 with unity increment. The table is generated in a new figure with 'N' on the column axes, GoS on the row axes and each column contains three internal columns of the three types; ErlangB, ErlangC and Binomial in order for better illustration of the compared traffic values.

3.4 Traffic comparator with number of trunks given range and GoS given values (Bonus):

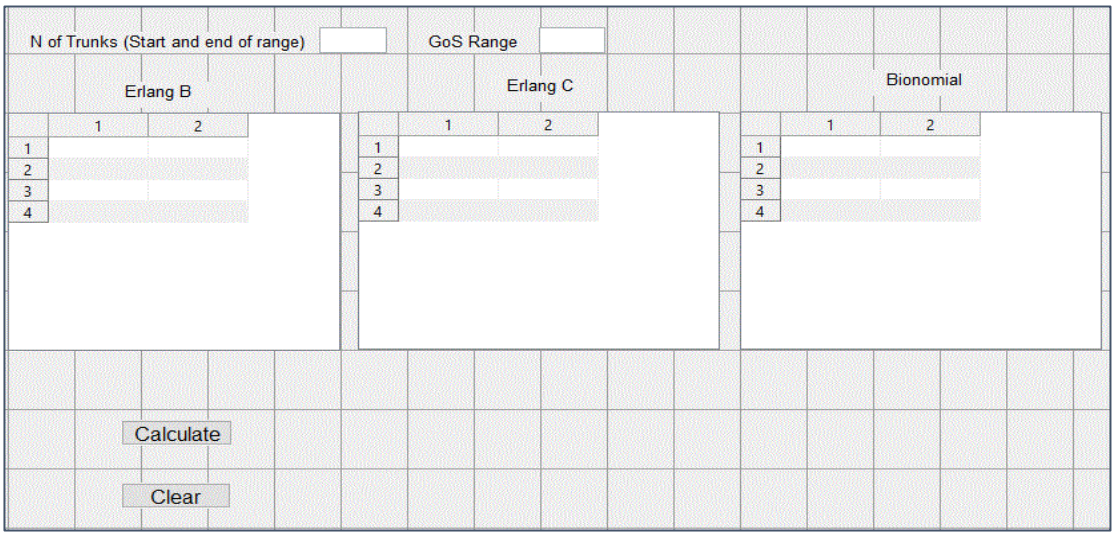


Figure 12 Traffic comparator with number of trunks given range and GoS given values (Bonus):

Selecting this requirement is almost the same as the previous one with addition degree of freedom for the user to enter the required number of trunks 'N' range with unity increments (start and end of the range only) i.e. 10 20-> 10,11,12…,20 and given GoS values for the comparison i.e. 0.1 0.2 1 5 10 ->> GoS = (0.1,0.2,1,5,10).

4. Output results

Computing the traffic given number of trunks (N) N ranges between 1 and 10 with unity increments and grade of service (GoS) takes the values (0.5%, 1%, 2%, 3%, 5%) and displaying the generated tables in section 3:

4.1 Part a: (Calculate traffic and GoS) "Erlang C"

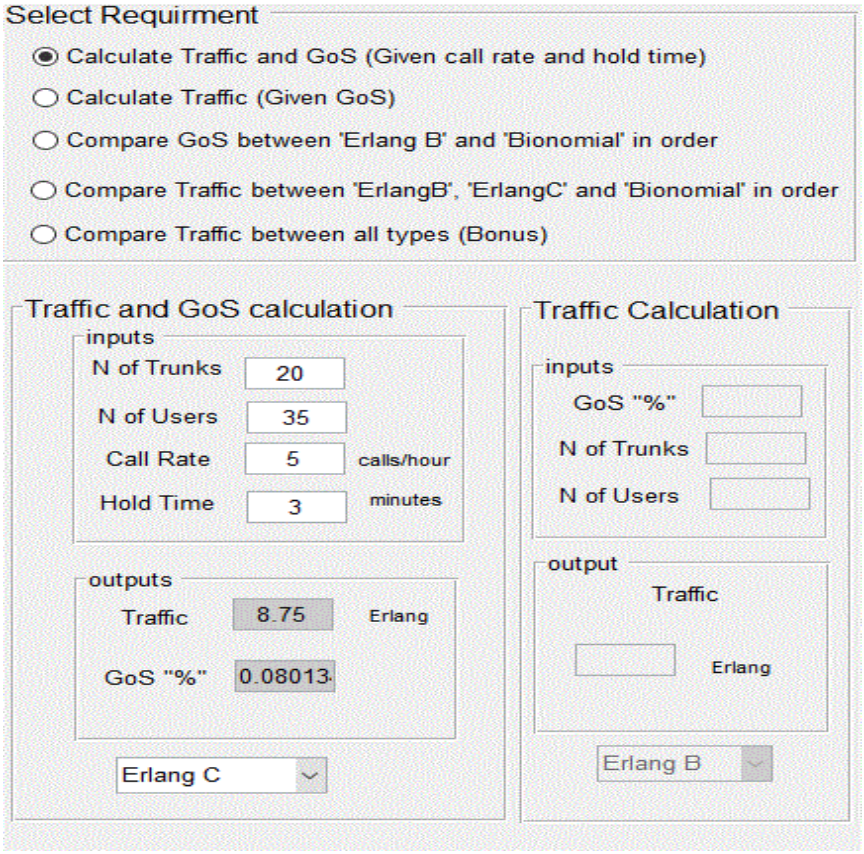


Figure 13 Part a: (Calculate traffic and GoS) "Erlang C"

4.2 Part b: (Calculate traffic (given GoS)) "ErlangB"

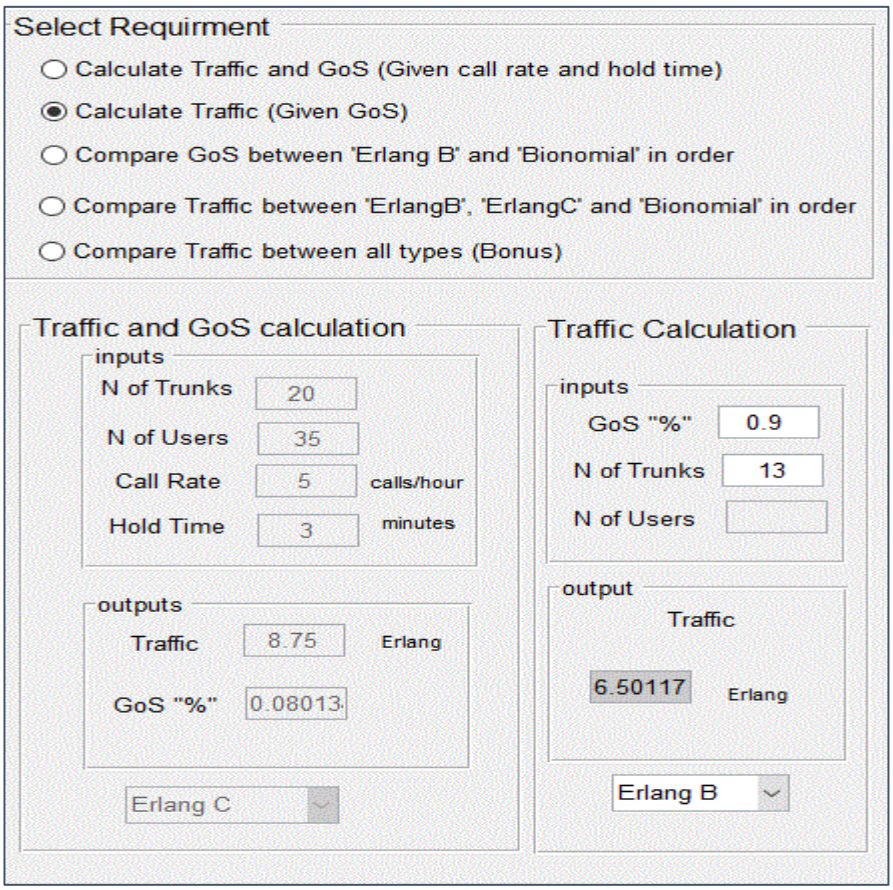


Figure 14 Part b: (Calculate traffic (given GoS)) "ErlangB"

4.3 Part a & part b "Binomial"

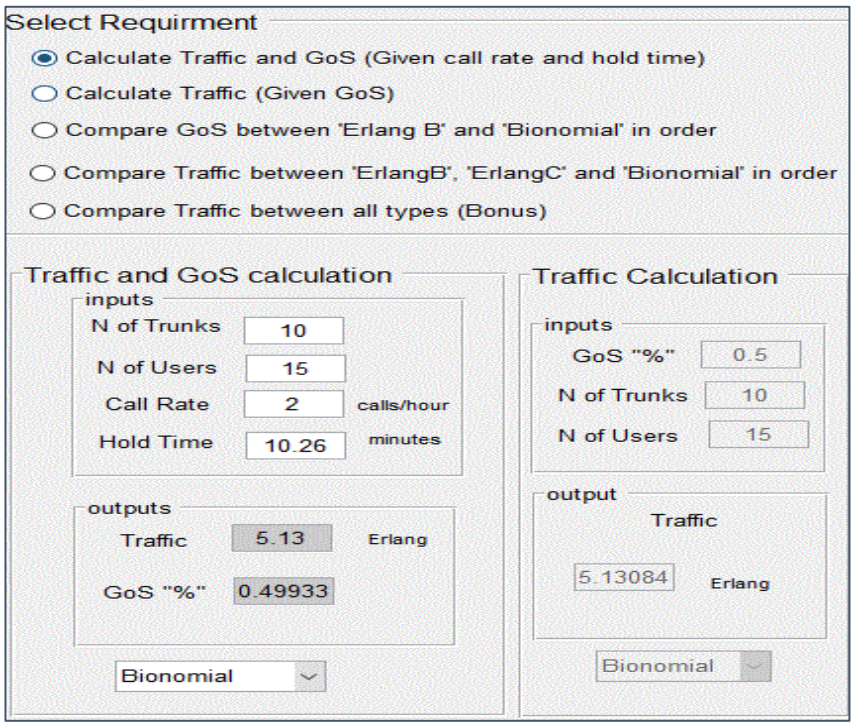


Figure 15 Part a & part b "Binomial"

4.4 GoS comparator:

	5		10		15		20		25		30		35		40		45		50	
1	50.0000	59.0400	66.6667	86.5782	75.0000	95.6020	80.0000	98.5588	83.3333	99.5278	85.7143	99.8453	87.5000	99.9493	88.8889	99.9834	90.0000	99.9946	90.9091	99.9982
2	20.0000	18.0800	40.0000	56.3792	52.9412	80.2088	61.5385	91.7134	67.5676	96.6943	72.0000	98.7234	75.3846	99.5183	78.0488	99.8214	80.1980	99.9347	81.9672	99.9764
3	6.2500	2.7200	21.0526	26.1802	34.6154	55.1949	45.0704	76.3111	52.9661	88.5483	59.0164	94.7968	63.7546	97.7404	67.5462	99.0519	70.6395	99.6128	73.2064	99.8452
4	1.5385	0.1600	9.5238	8.5642	20.6107	30.1810	31.0680	54.4911	39.8343	73.6138	46.9565	85.9620	52.7345	92.9994	57.4635	96.6793	61.3809	98.4861	64.6663	99.3316
5	0	0	3.6697	1.9581	11.0054	12.9840	19.9067	32.6712	28.4868	54.0123	36.0400	71.6054	42.4719	83.8137	47.9008	91.3409	52.4908	95.5990	56.3952	97.8551
6	0	0	1.2085	0.3066	5.2157	4.3854	11.7162	16.3062	19.1847	34.4108	26.4922	53.6596	33.1330	70.0351	38.9752	81.9987	44.0516	89.8248	48.4515	94.5328
7	0	0	0.3441	0.0314	2.1864	1.1610	6.2749	6.7600	12.0519	18.8929	18.5055	35.7139	24.8871	53.3860	30.8165	68.7639	36.1585	80.4417	40.9041	88.4420
8	0	0	0.0859	0.0019	0.8132	0.2397	3.0420	2.3278	7.0048	8.9171	12.1876	20.9727	17.8822	36.7369	23.5570	53.1658	28.9158	67.7075	33.8318	79.0883
9	0	0	0.0191	0.0001	0.2703	0.0382	1.3340	0.6658	3.7458	3.6175	7.5145	10.8382	12.2101	22.6892	17.3141	37.5677	22.4300	52.9836	27.3208	66.8115
10	0	0	0	0	0.0810	0.0046	0.5308	0.1579	1.8385	1.2621	4.3142	4.9264	7.8741	12.5437	12.1661	24.1359	16.7963	38.2597	21.4582	52.8296

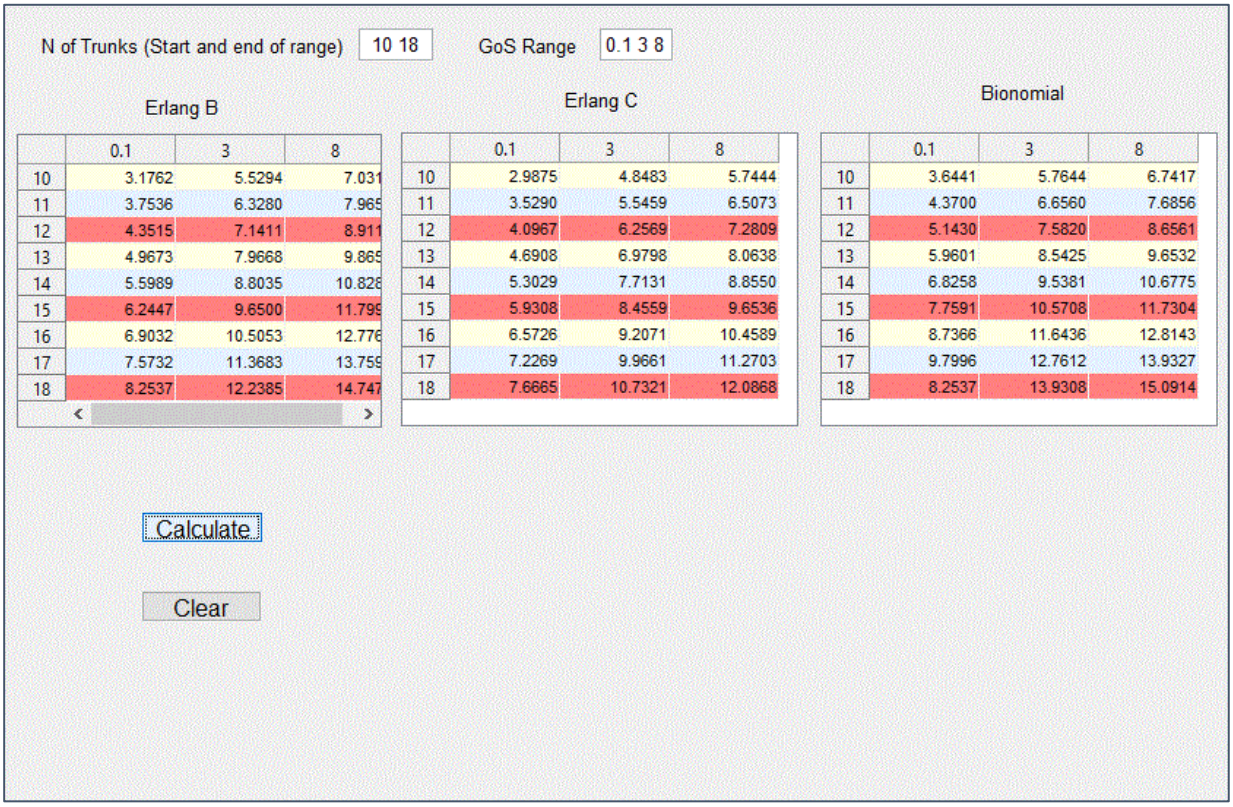
Figure 16 GoS comparator

4.5 Traffic comparator:

	0.1			0.5			1			5			10		
1	0.0010	0.0010	0.0011	0.0050	0.0050	0.0054	0.0101	0.0100	0.0108	0.0526	0.0500	0.0549	0.1111	0.1000	0.1125
2	0.0458	0.0454	0.0504	0.1054	0.1025	0.1146	0.1526	0.1465	0.1643	0.3813	0.3422	0.3900	0.5954	0.5000	0.5799
3	0.1943	0.1895	0.2187	0.3492	0.3340	0.3857	0.4555	0.4291	0.4959	0.8994	0.7876	0.9165	1.2708	1.0397	1.2222
4	0.4416	0.4266	0.5082	0.7021	0.6644	0.7888	0.8695	0.8100	0.9604	1.5246	1.3186	1.5607	2.0454	1.6531	1.9641
5	0.7675	0.7370	0.9032	1.1340	1.0658	1.2989	1.3609	1.2591	1.5289	2.2185	1.9052	2.2908	2.8811	2.3132	2.7770
6	1.1561	1.1057	1.3946	1.6251	1.5209	1.9008	1.9092	1.7585	2.1853	2.9603	2.5316	3.0911	3.7585	3.0066	3.6474
7	1.6051	1.5233	1.9786	2.1576	2.0190	2.5860	2.5017	2.2968	2.9208	3.7378	3.1882	3.9538	4.6663	3.7251	4.5683
8	2.0968	1.9802	2.6464	2.7302	2.5432	3.3515	3.1298	2.8663	3.7320	4.5430	3.8687	4.8754	5.5972	4.4633	5.5370
9	2.6225	2.4700	3.3960	3.3332	3.0996	4.1979	3.7866	3.4617	4.6196	5.3703	4.5688	5.8562	6.5465	5.2177	6.5534
10	3.1762	2.9875	4.2404	3.9617	3.6794	5.1308	4.4673	4.0789	5.5885	6.2157	5.2853	6.8999	7.5107	5.9855	7.6205

Figure 17 Traffic Comparator

4.6 Traffic comparator with number of trunks given range and GoS given values (Bonus):



	Erlang B 0.1	Erlang B 3	Erlang B 8	Erlang C 0.1	Erlang C 3	Erlang C 8	Bionomial 0.1	Bionomial 3	Bionomial 8
10	3.1762	5.5294	7.031	2.9875	4.8483	5.7444	3.6441	5.7644	6.7417
11	3.7536	6.3280	7.965	3.5290	5.5459	6.5073	4.3700	6.6560	7.6856
12	4.3515	7.1411	8.911	4.0967	6.2569	7.2809	5.1430	7.5820	8.6561
13	4.9673	7.9668	9.865	4.6908	6.9798	8.0638	5.9601	8.5425	9.6532
14	5.5989	8.8035	10.828	5.3029	7.7131	8.8550	6.8258	9.5381	10.6775
15	6.2447	9.6500	11.795	5.9308	8.4559	9.6536	7.7591	10.5708	11.7304
16	6.9032	10.5053	12.776	6.5726	9.2071	10.4589	8.7366	11.6436	12.8143
17	7.5732	11.3683	13.759	7.2269	9.9661	11.2703	9.7996	12.7612	13.9327
18	8.2537	12.2385	14.747	7.6665	10.7321	12.0868	8.2537	13.9308	15.0914

Figure 18 Bonus